(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 926 589 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
30.06.1999 Bulletin 1999/26

(51) Int. Cl.[6]: **G06F 9/00**, G06F 9/318

(21) Application number: **97830716.3**

(22) Date of filing: **24.12.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant:
STMicroelectronics S.r.l.
20041 Agrate Brianza (Milano) (IT)

(72) Inventors:
- **Pappalardo, Francesco**
  95047 Paterno' (Catania) (IT)
- **Tesi, Davide,**
  c/o SGS-Thomson Microelectronics
  01637 Saint-Genis-Pouilly Cedex (FR)
- **Mammoliti, Francesco Nino**
  88013 Dasa' (Vibo Valentia) (IT)
- **Bombaci, Francesco**
  98124 Messina (IT)

(74) Representative: **Botti, Mario**
Botti & Ferrari S.r.l.
Via Locatelli, 5
20124 Milano (IT)

(54) Processor having internal control instructions

(57) The present invention relates to a processor provided with a set of instructions formed, in general, of an operation section (S1) and an operand section (S2); for at least one (CNTR) of the instructions, the operand section (S2) represents operation control signals (CO) of the processor. In this way, an extension of the set of instructions can be simulated for tailoring the set of instructions to the user's own requirements. Consequently, the processor control unit (UC) should be capable of coupling its outputs (CO) to its inputs (II) upon receiving one such instruction, thereby to transfer such internal operation control signals without interpretation.

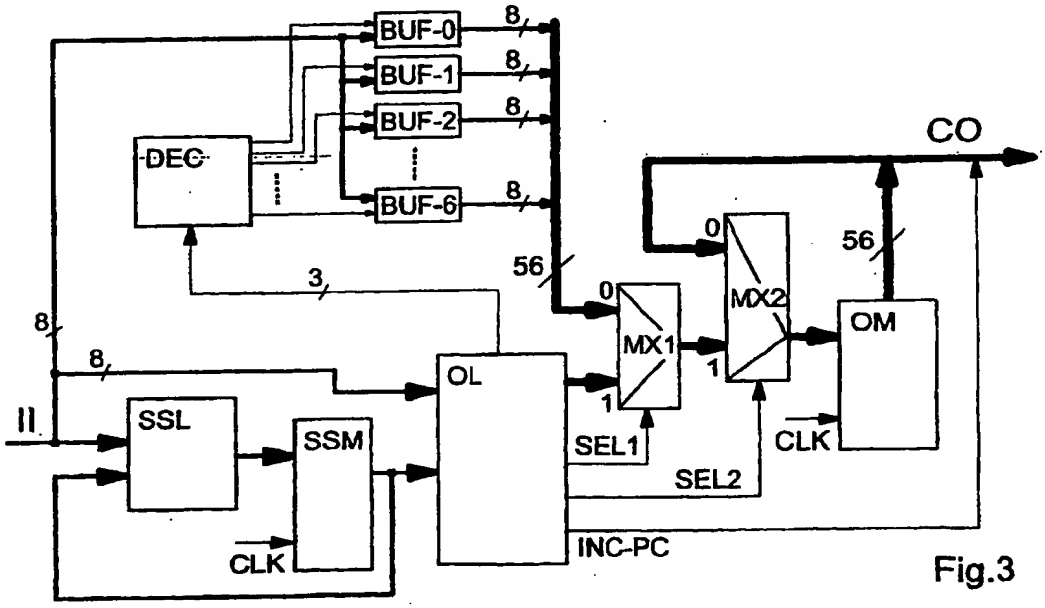

Fig.3

EP 0 926 589 A1

Description

[0001] This invention relates to a processor according to the preamble of Claim 1 or Claim 5.

[0002] A reference book on processor architectures is, for example, L. Ciminiera and A. Valenzano, "Advanced Microprocessor Architectures", Addison-Wesley, 1987, wherein both traditional and advanced architectures, such as CISC (Complex Instruction Set) and RISC (Reduced Instruction Set) configurations, are illustrated.

[0003] In fact, to enhance the calculating capabilities of processors, there are two opposite courses that can be followed: a first course consists of providing the processors with plural complex instructions, quite powerful but slow to execute, and the second consists of providing the processors with few simple instructions, less powerful but quickly executed.

[0004] An obvious solution is that of making each time the most convenient compromise (of instruction complexity versus speed of execution) for the user of the processor. However, this cannot be carried into effect exhaustively, and in consequence, different processors are offered on the market for different types of applications.

[0005] It is an object of this invention to solve the problem outlined above by providing a processor with a set of instructions which can be easily expanded and/or customized by the user himself.

[0006] This object is achieved by processors having the features defined in either Claim 1 or 5; further advantageous aspects of the invention are set forth in the subclaims.

[0007] The principle behind this invention is that of providing the processor with at least one control instruction wherein the operand section represents control signals for controlling the processor operation; in this way, an extension of the set of instructions can be simulated.

[0008] Accordingly, the control unit of the processor should be capable of coupling its outputs to its inputs, upon receiving an instruction as above, so as to transfer such internal operation control signals without any interpretation.

[0009] According to another aspect, the invention also relates to an integrated circuit and a processing system as defined in Claims 9 and 10, respectively, in which the processor can be advantageously included.

[0010] The invention can be better understood by having reference to the following description, to be read in conjunction with the accompanying drawings, in which:

Figure 1 shows schematically a block diagram of a prior art processor;

Figure 2 is a partial diagram of a set of instructions in a processor according to the invention;

Figure 3 is a block diagram of a control unit in a processor according to the invention; and

Figure 4 shows schematically a state transition diagram of the control unit in Figure 3.

[0011] Considering the example of Figure 1, a prior art processor comprises a plurality HPD of operating blocks which have operation control inputs CO, current address outputs CA and current data inputs/outputs CD, and a control unit UC having instruction inputs II and control outputs CO connected to said control inputs CO; provided within the processor are a control bus BC, a data bus BD, and an address bus BA, in general all bidirectional. These internal buses are connected to corresponding external buses, with the control bus BC being connected to the control unit UC, the data bus BD connected to the instruction inputs II and the current data I/O's CD, and the address bus BA connected to the current address outputs CA.

[0012] The plurality HPD of operating blocks include, for example, an accumulator register ACC, a program count register PC, a few working registers (of which two, REG1 and REG2, are shown), an arithmetic and logic calculating unit ALU, and a timing unit TIM. These operating blocks are each provided with one or more operation control inputs (collectively designated CO), and connected to the data I/O's CD and/or the address outputs CA according to the operations that they are to perform.

[0013] The basic duty of the control unit UC is to interpret instructions received on its inputs II, and consequently generate, on its outputs CO, suitable internal operation control signals for the operating blocks. The duty of the control bus BC and the meaning conveyed by the external control signals being propagated therethrough will be no further discussed herein because foreign to this invention and well known from literature.

[0014] Referring to Figure 2, the set of instructions of the processor in Figure 1 may include, for example, the instructions ADD, LOAD, INC-ACC, JMP-REL, JMP-ABS, and many more. These instructions are formed of an operation section S1 of fixed length -- e.g. 1 byte (8 bits) of an operational code OPC -- and an operand section S3 of varying length -- e.g. 1, 2, 3 data DAT or address ADR bytes, or bytes of a generic operand OP -- excepting the instruction INC-ACC which requires no operand, as explained hereinafter.

[0015] The instructions convey the following meanings:

ADD -- add together the contents of the register corresponding to the first operand OP1 and the contents of the register corresponding to the second operand OP2, and place the result into the accumulator register;

LOAD -- load the memory contents identified by the corresponding address to the operand ADR, into the accumulator register;

INC-ACC -- increase the contents of the accumulator register;

JMP-REL -- jump to executing the instruction included in the memory word identified by the address being the sum of the program count register PC contents and the operand DAT;

JMP-ABS -- jump to executing the instruction included in the memory word identified by the corresponding address to the operand ADR.

[0016] In the processor of this invention, there is at least one instruction, designated CNTR in Figure 2 and being formed of an operation section S1 and an operand section S2, wherein the operand section, corresponding to 7 bytes in Figure 2, represents internal (and external, if any) operation control signals of the processor.

[0017] With the instruction CNTR, any new operation can be performed (as allowed by the operating blocks and the connections inside the plurality HPD), using as the operand data corresponding to values of the control signals which would implement that new operation. For example, using an instruction CNTR, the registers REG1 and REG2 can be loaded simultaneously from the memory. The step of acquiring an instruction CNTR would obviously take longer than the steps for the other instructions (in the example of Figure 2, 6 bytes must be acquired instead of 4 bytes at most), but is compensated by the almost complete absence of instruction interpreting activity.

[0018] In this way, each user is enabled to add specific instructions to the standard set of instructions, as the user's particular application may require.

[0019] This extension is achieved neither at the expense of the efficiency of standard instruction execution, nor of circuit complexity of the processor instruction interpreter.

[0020] Of the various viable courses, the simplest implemented is one where the operand section would represent all the (typically) internal processor operation control signals.

[0021] In this case, if the control of the program execution flow is to be taken away from the programmer, then it is more convenient to have the operand section represent all the internal control signals of the processor operation, but for control signals to the program count register PC; the register PC would then be managed conventionally by the control unit UC.

[0022] Alternatively, a number of instructions may be provided whose respective operand sections represent discrete sets of (typically) internal processor operation control signals; thus, the lengths of the control instructions can be greatly reduced. These sets may be separate or partly cross one another

[0023] Referring to Figure 3, the control unit UC may be a finite state machine in a conventional processor, and may include proximate state logic circuitry SSL having first inputs connected to the inputs II of the unit UC, a state memory SSM having inputs connected to the outputs of the circuitry SSL and outputs connected to second inputs of the circuitry SSL, as well as output logic circuitry OL having first inputs connected to the inputs II of the unit UC and second inputs connected to the outputs of the memory SSM; the memory SSM also has an input CLK for a clock signal. The outputs of the circuitry OL may either be connected to the outputs CO of the unit UC directly, or through a latching arrangement as shown in Fgure 3; the latching arrangement would comprise a multiplexer MUX2 having first inputs coupled to the outputs of the circuitry OL and having a selection input SEL2 connected to a particular output of the circuitry OL, an output memory OM having inputs connected to the outputs of the multiplexer MUX2 and outputs connected to both the outputs CO and to the second inputs of the multiplexer MUX2; the memory OM also has an input CLK for a clock signal.

[0024] Alternatively, the control unit could be microprogrammed.

[0025] In the processor of this invention, a circuit section (DEC, BUF-0...BUF-6, MUX1) is added in the control unit UC for coupling the outputs CO to the inputs II such that information can be transferred from the inputs to the outputs.

[0026] Since, in general, the number of outputs CO -- 56 in the example of Figure 3 -- would be much larger than the number of inputs II -- 8 in the example of Figure 3 -- it may be arranged for the control unit UC to include buffer logic circuitry BUF-0...BUF-6; the coupling of the inputs II and outputs CO may then be established through this circuitry such that the transferrment of information from the inputs II to the circuitry BUF will take place at successive time phases, and the transferrment of information from the circuitry BUF to the outputs CO at one time phase.

[0027] In order to reduce the length of the control instructions CNTR, it could be conceived of encoding the control signals. For example, if the calculating unit ALU can effect eight different arithmetic and logic operations, and has eight corresponding operation control inputs available, three bits encoded in the instruction CNTR would be sufficient. In this case, the control unit UC would require decoding logic circuitry (omitted from the example in Figure 3), and the inputs II and outputs CO would be coupled through this circuitry.

[0028] Of course, the control unit UC must be capable of discriminating between the control instruction CNTR and the other instructions; for the purpose, it includes an instruction interpreter (SSL, SSM, OL) arranged to interpret a set of instructions of which at least one is formed of an operation section and an operand section, the operand section representing values of operation control signals.

[0029] In the embodiment of Figure 3, the 8 inputs II are connected in parallel to 7 buffers BUF-0, BUF-1,

BUF-2, BUF-3, BUF-5, BUF-6; the outputs of the buffers BUF are connected to 56 first inputs of a multiplexer MUX1; 56 second inputs of the multiplexer MUX1 are connected to the outputs of the circuitry OL, and one selection input SEL1 of this multiplexer is connected to a particular output of the circuitry OL; the outputs of the multiplexer MUX1 are connected to the first inputs of the multiplexer MUX2; the buffers BUF also have inputs CLK (omitted from Figure 3 for simplicity) for a clock signal, and activation inputs respectively connected to a decoder DEC. The decoder DEC has three inputs connected to particular outputs of the circuitry OL, and eight outputs, of which one is not used and is omitted from Figure 3. It should be noted that the circuitry OL is provided with a particular separate output for the increase control signal INC-PC to the program count register PC.

[0030] The operation of the control unit UC is more clearly explained with the aid of the state transition diagram of Figure 4.

[0031] At startup, the unit is in the state ST00.

[0032] Upon receiving an instruction CNTR, the unit goes over to the state ST10, SEL1 is set to "0", SEL2 is set to "0", DEC is set to "000", and INC-PC is set active; the memory is now addressed.

[0033] Upon a clock pulse, the unit goes over to the state ST11, SEL1 and SEL2 and DEC remain stable, and INC-PC is set inactive; the first byte is now stored into the buffer BUF-0.

[0034] Upon a clock pulse, the unit goes over to the state ST10, SEL1 is set to "0", SEL2 is set to "0", DEC is set to "001", and INC-PC is set active; the memory is now addressed.

[0035] The states ST10 and ST11 are reiterated until all the buffers BUF are loaded with data; thereafter, the unit will go over to the state ST12, SEL will be set to "0", SEL2 set to "1", and INC-PC set active; the control signals are now supplied to the outputs CO and preparations are made for acquiring the next instruction from the memory.

[0036] Upon a clock pulse, the unit is restored to its initial state ST00, SEL2 is set to "0", and INC-PC is set inactive; the operating code OPC of the instruction, as just acquired, is then decoded.

[0037] Upon a clock pulse, and dependent on the result of the decoding operation, the unit goes over to either the state ST10 or the state ST20.

[0038] When an ordinary instruction is received, the unit goes over to the state ST20, SEL1 is set to "1", SEL2 is set to "0", and INC-PC is set active; the memory is now addressed.

[0039] Upon a clock pulse, the unit goes over to the state ST21, SEL1 and SEL2 remain stable, and INC-PC is set inactive; the first byte of the operand is presently acquired.

[0040] Upon a clock pulse, the unit goes over to the state ST20, SEL1 is set to "1", SEL2 is set to "0", and INC-PC is set active; the memory is presently addressed.

[0041] The states ST10 and ST11 are iterated until the instruction is fully acquired; subsequently to this, the unit will go over to the state ST22, SEL1 will be set to "1", SEL2 set to "1", and INC-PC set active; thus, the control signals are supplied to the outputs CO, and preparations are made for acquiring the next instruction from the memory.

[0042] Upon a clock pulse, the unit is restored to its initial state ST00, SEL2 is set to "0", and INC-PC is set inactive; the operating code OPC of the instruction just acquired is then decoded.

[0043] The above repeats itself throughout the processor operation.

[0044] It will be appreciated that this processor may be connected to advantage in a semiconductor integrated circuit, or in a single- or multi-processor type of processing system.

Claims

1. A processor arranged to execute instructions from a predetermined set of instructions of which at least one is formed of an operation section (S1) and an operand section (S2), characterized in that it is provided with at least one instruction (CNTR) whose operand section (S2) represents control signals (CO) controlling the processor operation.

2. A processor according to Claim 1, provided with an instruction whose operand section represents all the internal control signals of the processor operation.

3. A processor according to Claim 1, provided with an instruction whose operand section represents all the internal control signals of the processor operation, excepting signals (INC-PC) controlling the program counter register (PC).

4. A processor according to Claim 1, provided with a number of instructions whose respective operand sections represent distinct sets of internal control signals of the processor operation.

5. A processor of the type which comprises:

   a) a plurality (HPD) of operating blocks (REG1, REG2,ACC,PC,ALU,TIM) having operation control inputs; and

   b) a control unit (UC) having instruction inputs (II), and having control outputs (CO) connected to said control inputs, said unit being adapted to interpret instructions received on its inputs and to generate, in consequence, at its outputs operation control signals of said operating blocks (REG1,REG2,ACC,PC,ALU,TIM);

characterized in that said control unit (UC) is also adapted to couple its outputs (CO) to its inputs (II) for transferring information from the inputs to the outputs.

6. A processor according to Claim 5, wherein said control unit (UC) comprises buffer logic circuitry (BUF-0,...,BUF-6), and wherein said coupling is effected through said circuitry (BUF) such that the transfert of information from the inputs (II) to the circuitry (BUF) will take place at successive time phases and the transfert of information from the circuitry (BUF) to the outputs (CO) will take place at a single time phase.

7. A processor according to Claim 5, wherein said control unit comprises decoding logic circuitry, and wherein said coupling is effected through said circuitry.

8. A processor according to Claim 5, wherein said control unit comprises an instruction interpreter arranged to interpret a set of instructions of which at least one (CNTR) is formed of an operation section (S1) and an operand section (S2), the operand section (S2) representing values of said operation control signals (CO).

9. A semiconductor integrated circuit including a processor according to one or more of the preceding claims.

10. A processing system including at least one processor according to one or more of the preceding claims.

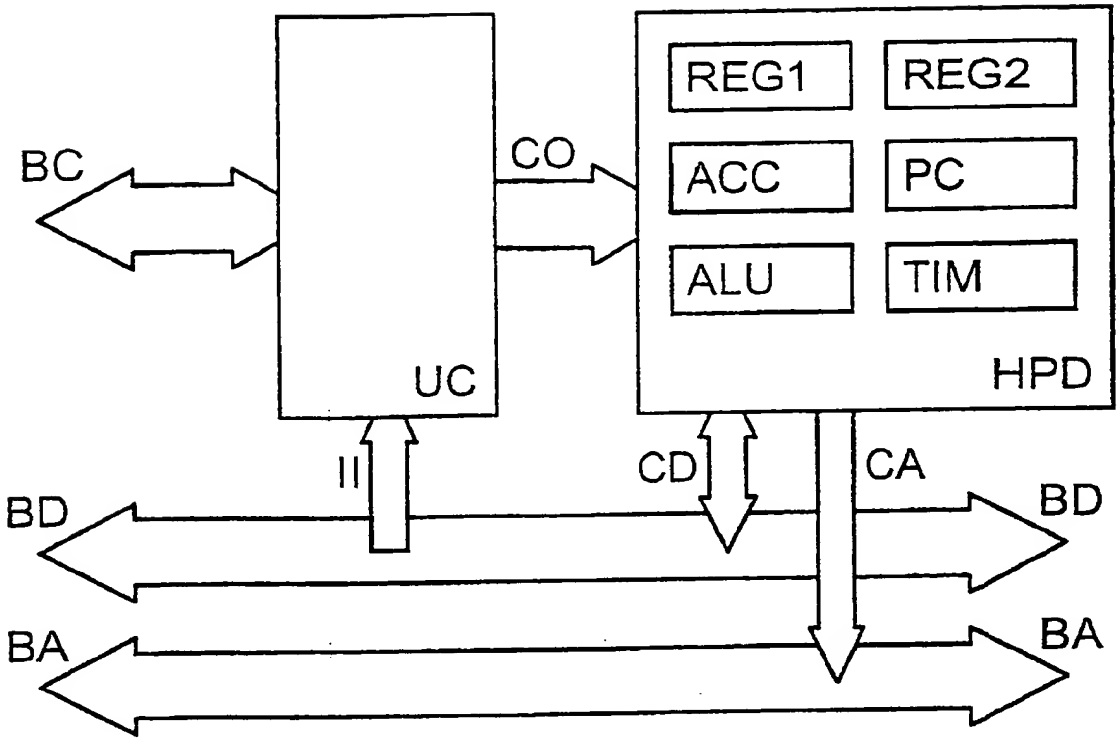
REG1
REG2
ACC
PC
ALU
TIM
HPD
BC
CO
UC
II
CD
CA
BD
BD
BA
BA

Fig.1

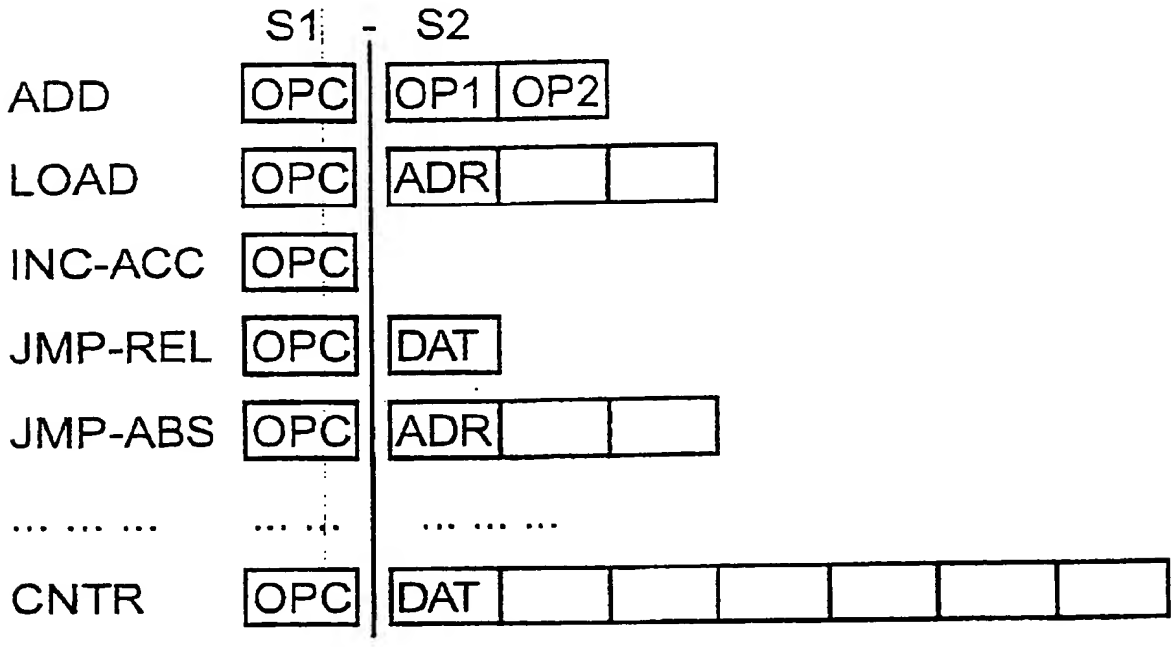
S1
S2
ADD
OPC
OP1
OP2
LOAD
OPC
ADR
INC-ACC
OPC
JMP-REL
OPC
DAT
JMP-ABS
OPC
ADR
CNTR
OPC
DAT

Fig.2

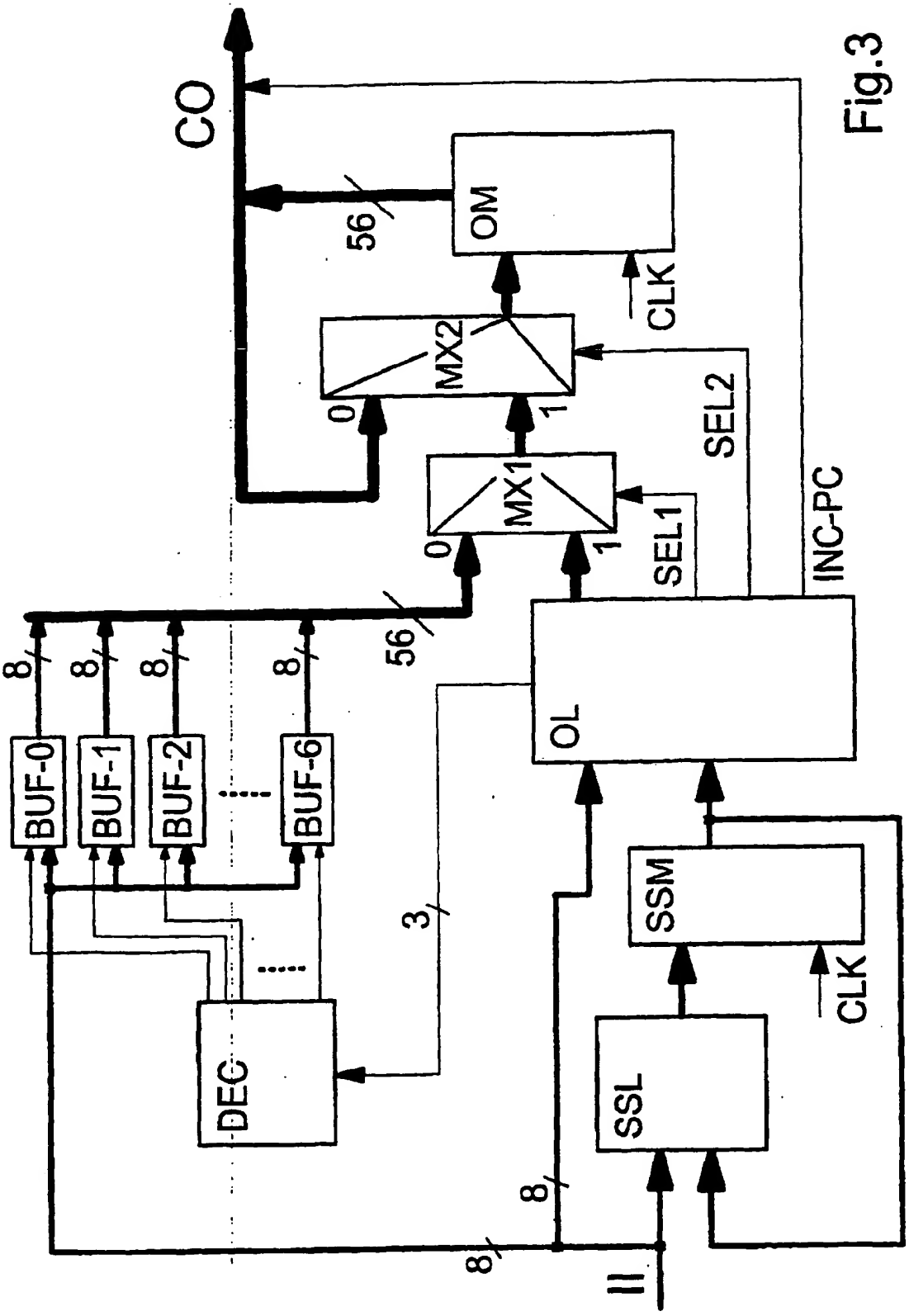
CO
56
OM
CLK
MX2
0
1
SEL2
MX1
0
1
SEL1
INC-PC
56
8
8
8
8
BUF-0
BUF-1
BUF-2
BUF-6
OL
3
DEC
SSM
CLK
SSL
8
8
II

Fig.3

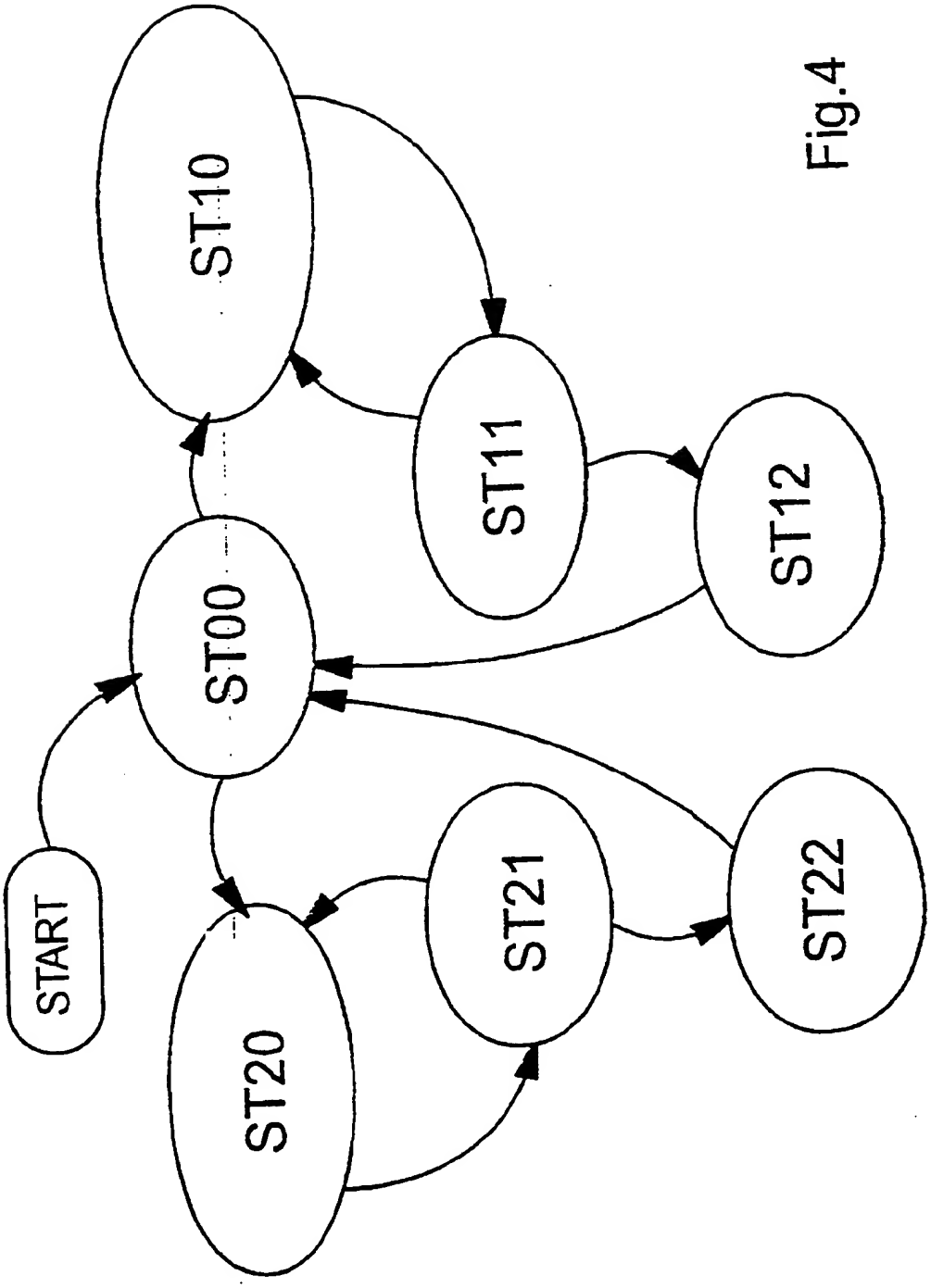

Fig.4

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 97 83 0716

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
Y	BÉRAUD AND ESTEBAN: "Microprocessor operation code expander" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 20, no. 12, May 1978, ARMONK,US, pages 5197-5199, XP002065481	1,2,4	G06F9/00 G06F9/318
A	* the whole document *	8	
Y	"SELECTING PREDECODED INSTRUCTIONS WITH A SURROGATE" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 36, no. 6A, 1 June 1993, pages 35-38, XP000370750	1,2,4	
A	* the whole document *	8	
A	FORSELL M J: "MINIMAL PIPELINE ARCHITECTURE - AN ALTERNATIVE TO SUPERSCALAR ARCHITECTURE" MICROPROCESSORS AND MICROSYSTEMS, vol. 20, no. 5, September 1996, pages 277-284, XP000639809 * page 279, right column, section 3.3 *	1,2	TECHNICAL FIELDS SEARCHED (Int.Cl.6)
A	WO 91 11765 A (TERAPLEX INC) * summary; page 9, line 28 - page 10, line 28 *	1,2,5	G06F
A	DIRAC J F: "CONTROL WORD EXPANSION" IBM TECHNICAL DISCLOSURE BULLETIN, vol. 3, no. 7, December 1960, page 23 XP002054504 * the whole document *	6	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	20 May 1998	Klocke, L

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 97 83 0716

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

20-05-1998

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9111765 A	08-08-91	AU 7305491 A	21-08-91
		US 5675777 A	07-10-97

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

A High-Speed RISC CPU Using the QL16x24 FPGA

Bruce Kleinman and Bill Cox
QuickLogic Corporation
Behringstrasse 10
D-82152 Planegg, Germany
49 (89) 899 143-28

Traditionally, FPGAs have been employed to provide glue logic in the form of datapath, state machine, and arithmetic circuits. Today's FPGAs tackle entire sub-systems at increasing clock rates and increasing levels of complexity. Leading-edge applications include support for today's fastest microprocessors, 2- and 3-dimensional graphics computation, data compression and expansion, error correction, and video and imaging processing.

State-of-the-art FPGAs will open new application areas for programmable logic previously possible only with ASIC or custom solutions. New submicron programmable ASICs (pASICs) from QuickLogic will push the industry's fastest FPGA architecture to higher speeds and higher densities. This paper describes an intriging application— a high-performance 16-bit RISC CPU designed in a QuickLogic QL16x24 FPGA.

pASIC Architecture

The pASIC architecture was introduced in 1991 with the pASIC-1 Family of devices. The architecture consists of a regular array of Logic Cells connected by a highly orthogonal routing structure. Shown in Figure 1, the Logic Cell has 23 inputs and 5 outputs.

The high fan-in allows 14-input gates to be implemented in a single Logic Cell, greatly reducing the number of cell delays required to provide wide gating functions. The multiple outputs allows the cell to be broken into fragments for efficient use when wide gating functions are not required (QuickLogic's tools automatically map as many as six separate macros into a single Logic Cell).

The wide fan-in, the multiple outputs, and the integrated flip-flop result in a fast and flexible building block. The pASIC routing structure is unique in three regards—the routing interconnect is very fast, it is very plentiful, and it is very orthogonal. The first two features are attributed to the ViaLink antifuse, which combines extremely low impedance with extremely small size.

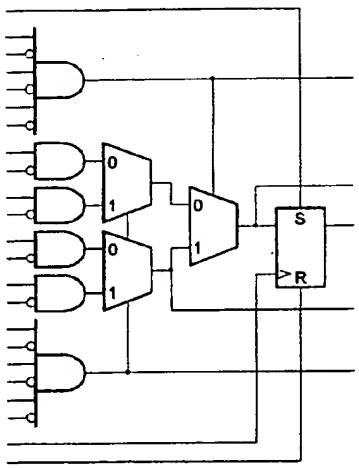

Figure 1. The pASIC Logic Cell

The extremely low impedance (20 to 50 ohms) of the ViaLink element produces net delays in the range of just 1.0 to 2.5 ns. Unlike other FPGAs, pASICs incur very little

delay in the routing interconnect; like a masked gate array, most of the delay is incurred doing useful work.

The extremely small size (less than 1.0 micron^2, effectively the same size as a standard M1-to-M2 via) of the ViaLink element permits a tremendous amount of wire—there are 26 wires in the vertical channels between logic cells and 12 wires in the horizontal channels between logic cells. The wire is not only abundant, it is also highly orthogonal—there is a ViaLink at every vertical-horizontal wire crossing. The practical result of this is 100% automatic place and route, even on designs that use 100% of the Logic Cells and 100% of the I/O pads.

The pASIC-1 Family has established its leadership position as the highest-performance FPGA architecture based on a 1.2 micron CMOS process. These parts, the QL 8x12A and QL 12x16, are used in real-world applications at speeds of 66 to 100 MHz; they benchmark at speeds of over 120 MHz.

In 1993 the family will be ported to a 0.65 micron CMOS process. These parts will be designed into applications at speeds of 80 to 130 MHz; they will benchmark at speeds of over 160 MHz.

Device	Usable Gates	Available Gates
QL 8x12A	1000	3000
QL 12x16	2000	6000
QL 16x24	4000	12000

Table 1. The pASIC-1 Family

Table 1 lists the devices in the pASIC-1 Family. The first two devices are presently fabricated in 1.2 micron CMOS; all three devices will be fabricated in 0.65 micron CMOS.

Datapath Circuits

Many applications require datapath circuits such as bus-multiplexers, registers and register files, and shifters. The pASIC Logic Cell is a very versatile building block for these circuits. Consider the circuit fragment shown in Figure 2.

The AND gates provide decode for the select lines on the multiplexer. The four data inputs on the multiplexer can be configured with any combination of inversion bubbles. The flip-flop provides both preset and clear. The Logic Optimizer integrated into SpDE (the Seamless pASIC Design Environment) will map this entire circuit fragment into a single Logic Cell.

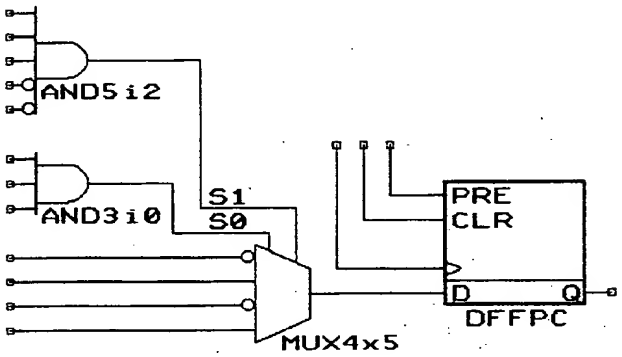

Figure 2. Datapath Example

An autonomous linear feedback shift register (ALFSR) can be constructed using a similar circuit fragment. An ALFSR produces a pseudo-random or pseudo-noise sequence that is employed in random circuit testing. A 24-bit ALFSR can be constructed in exactly 24 Logic Cells; this circuit operates at more than 150 MHz.

State Machines

State Machines form the heart of many applications—DRAM and DMA controllers,

cache controllers, bus arbitrators, for example. Numerous techniques are employed to implement state machines; two of the most popular are encoded state and one-hot encoding.

The encoded state technique was popularized by PLA and EPLD devices over the past 20 years. This technique employs a register that encodes the states. A 4-bit state register, for example, can encode as many as 16 states. The state register is fed by random logic that determines the state transitions based on the current state bits and the inputs to the state machine. This random logic is well suited to an AND-OR array, found at the heart of all EPLD macro cells.

Because the encoded state technique exploits wide fan-in gating functions, it typically challenges FPGA devices. These big gates often require four or five levels of low fan-in FPGA cells, resulting in large and slow implementations.

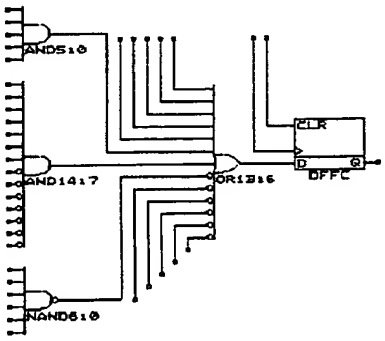

Figure 3. Encoded State Example

The pASIC Logic Cell is well suited to provide the wide gating functions required to build dense and fast encoded state machines. In one set of Logic Cells, wide AND gates (up to 14 inputs) are used to provide the product terms. In a second set of Logic Cells, wide OR gates (up to 13 inputs) are used to generate the sum terms that feed the state bits.

Figure 3 shows a circuit fragment from an encoded state machine, which requires only two levels of Logic Cells (the OR gate and the flip-flop will be packed automatically into a single Logic Cell). Note the inversion bubbles on six of the OR gate inputs; SpDE's Logic Optimizer will automatically "bubble-push" wherever necessary so that the designer need not worry about balancing the number of non-inverted and inverted signals. Customers have designed many state machines using this method, running at speeds of up to 80 MHz.

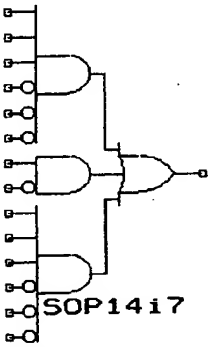

Figure 4. Sum-of-Products Macro

The pASIC Macro Library contains a sum-of-products macro, shown in Figure 4, which can be used to implement small encoded state machines in a single level of Logic Cells. Small state machines using this macro have been designed to run at speeds of up to 120 MHz.

The one-hot encoding technique (also known as bit-per-state encoding) was popularized by gate array and FPGA devices over the past 10 years. This technique employs one flip-flop for each state. In a manner of thinking, one-hot directly implements the state transition diagram by substituting a flip-flop for each state bubble.

A simple state diagram is shown in Figure 5. Although this example includes only three states (S1, S2, S3) and two input signals (R, T), the one-hot method scales well to very large state machines.

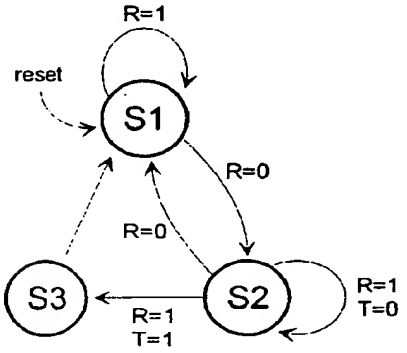

Figure 5. State Diagram

The implementation of this state machine is shown in Figure 6. Each state flip-flip and its driving logic function will be packed automatically into a Logic Cell, resulting in a single level of Logic Cells. This small example requires a total of three Logic Cells and runs at 160 MHz. Real-world customer one-hot state machines have been designed at speeds of up to 140 MHz.

One rule-of-thumb for selecting between these techniques is based on the ratio of state transitions to states. The encoded state technique is very good for state machines with a high ratio of transitions-to-states, while the one-hot technique is very good for state machines with a low ratio of transitions-to-states. The pASIC architecture is uniquely suited to implement both of these techniques efficiently, yielding dense and fast results.

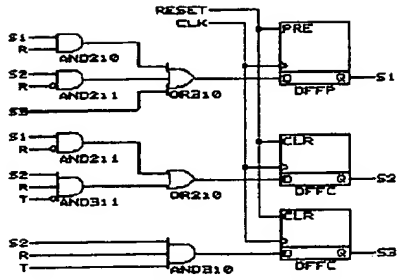

Figure 6. One-Hot Example

Arithmetic Circuits

While not as universal as state machines, many applications include arithmetic circuits. FPGAs are used to implement adders, multipliers, and comparators. Conventional logic holds that arithmetic favors a very fine grained FPGA cell. The multiple outputs of the pASIC Logic Cell permit very efficient and very fast arithmetic circuits.

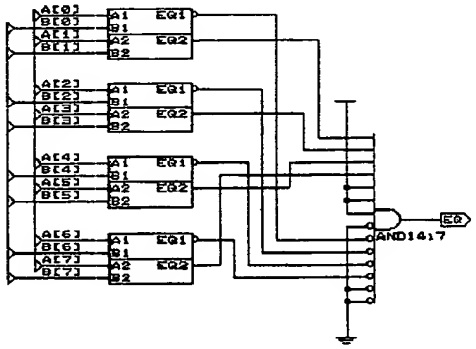

Figure 7. 8-bit Equality Comparator

The equality comparators in the pASIC Macro Library take advantage of the multiple outputs and the high fan-in of the Logic Cell.

The 8-bit equality comparator shown in Figure 7 requires a total of five Logic Cells, and features a propagation delay of 6 ns.

The conditional sum addition technique is well suited to multiplexer-based FPGAs. This technique is employed to build adders, subtractors, and accumulators in the pASIC Macro Library [see QuickNote QAN4, Fast Accumulators, for complete details]. The 8-bit accumulator, for example, requires only 20 Logic Cells and runs at over 90 MHz.

A 16-Bit RISC CPU

All of the circuits discussed—datapath, state machine, and arithmetic—are used in real-world applications. All three of these circuit types can be combined into an intriguing application—a high-performance 16-bit RISC CPU.

This CPU consists of a streamlined RISC core, a register file, an arithmetic logic unit (ALU), an on-chip instruction cache, and a memory management unit (MMU). A full Harvard architecture is employed (separate instruction and data busses off chip to memory). A block diagram of the CPU is shown in Figure 8.

The RISC core consists of the Control Unit and the Programmer Counter unit. A 16-bit instruction word is used, with a variable length op-code. This is somewhat of a break with the RISC tradition of fixed-length op-codes. It allows very efficient instruction encoding, however, and the instruction decode is still well out of the critical path.

The core uses a five stage pipeline—

Stage 1. Instruction Fetch
Stage 2. Instruction Decode
Stage 3. ALU-1
Stage 4. ALU-2 and Memory
Stage 5. Write Back

This pipeline exploits the parallelism in the CPU, allowing as many as five instructions to be processed simultaneously (this requires that all instructions hit in the on-chip cache).

The admittedly small on-chip instruction cache allows the core to operate at full-speed on loops. There is a single cycle penalty for a miss in the on-chip cache.

The Datapath Unit includes a small register file and a fairly full-featured ALU. The register file consists of four general purpose 16-bit registers. The size of the register file was chosen to minimize the multiplexer circuitry required to select the two registers feeding the ALU. All ALU operations are register to register.

The CPU is implemented in a single QL 16x24 device. Unlike some of its contemporaries, this CPU reached its target clock frequency of 66 MHz. Furthermore, the design was automatically placed and routed (SpDE's Timing-Driven Placer was used to optimize the placement of logic along the critical paths).

The current implementation leaves much of the QL 16x24 device unused. Future implementations will include on-chip peripherals such as a high-speed UART (an on-chip IEEE-compatible floating point unit, a long sought after goal, has been abandoned as impractical).

As QuickLogic is a hardware company, no assembler or compiler has been developed. Test programs have been written the way all software was meant to be written—directly in machine language.

Summary

Five years ago, FPGAs featured at most 2000 usable gates. Today's FPGAs offer up to 10,000 usable gates. This five-fold increase in capacity was not accompanied by a five-fold increase in performance. This discrep-

ancy between capacity and performance has limited FPGA applications.

The pASIC architecture was crafted to achieve high-performance with flexibility and scalability. The pASIC-1 Family of devices quickly established itself as the high-performance leader, providing this performance is across a broad range of applications. Accompanied by state-of-the-art design tools, the pASIC-1 devices combine speed and flexibility with ease of use.

The third member, the QL 16x24, improves upon the leadership performance of the pASIC-1 Family while providing higher capacity. This device will open up new applications for FPGAs by providing the performance and capacity normally associated with a genuine gate array.

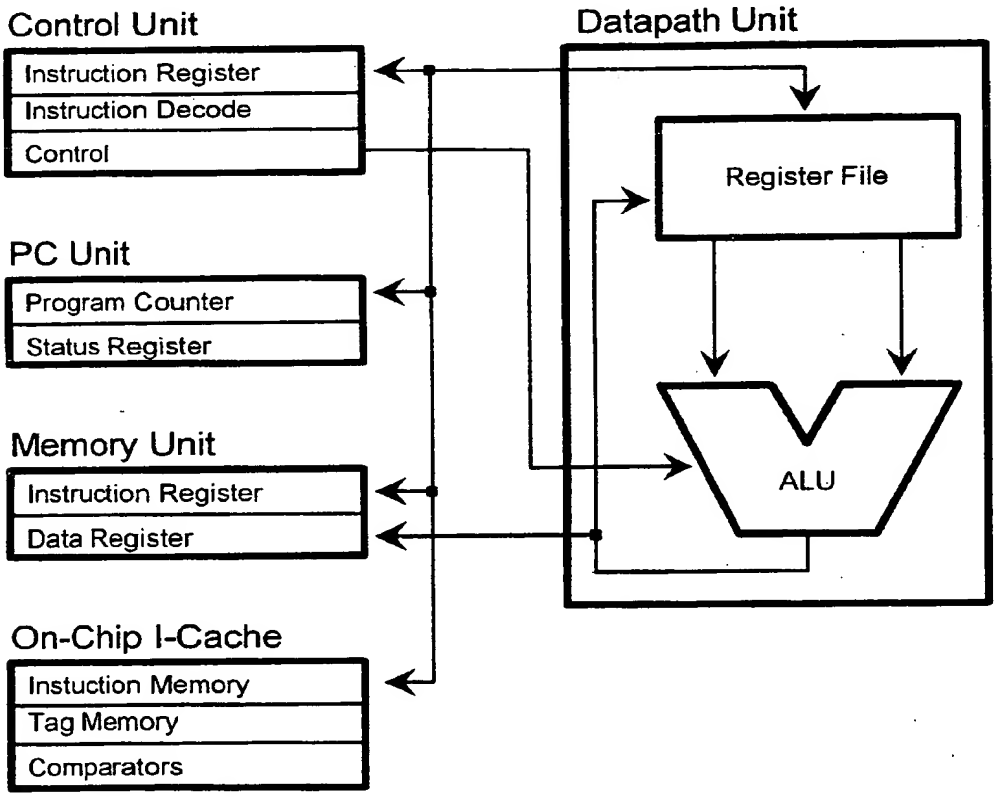

Figure 8. 16-bit RISC CPU Block Diagram